ROS and OpenCv for beginners | Blob Tracking and Ball Chasing with Raspberry Pi

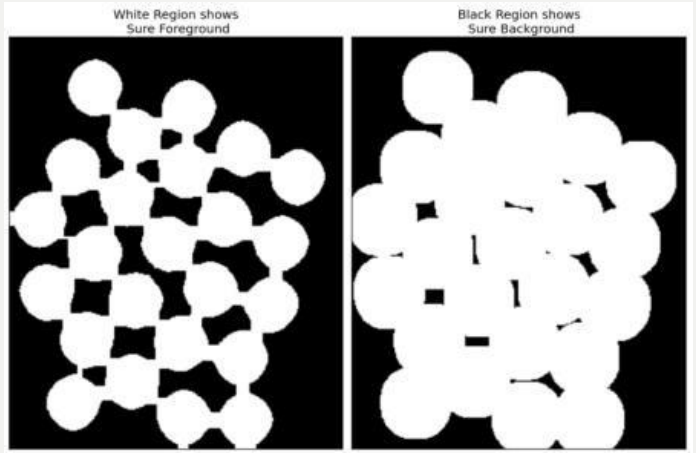
Opencv – Robot preception

Opencv

- 2D Image Processing
- Image Input/Output
- Video Input/Output
- Camera Calibration
- Video Analysis (motion extraction, feature tracking, foreground extraction, …)
- Object Detection
- Machine Learning, Deep Neural Networks
- GPU-Accelerated Computer Vision
- and much more …

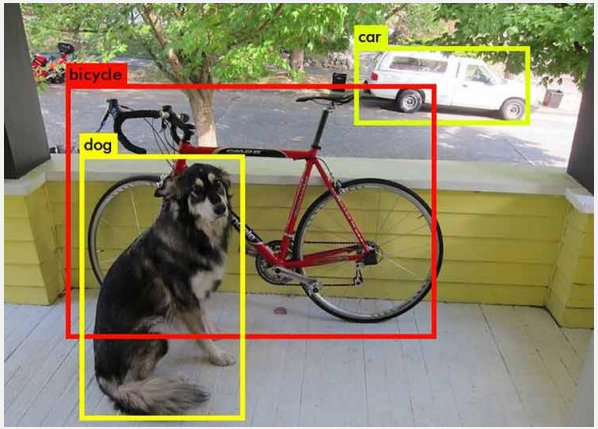
IMAGE SEGMENTATION

- the process of partitioning a digital image into multiple segments
- used to locate objects and boundaries (lines, curves, etc.) in images

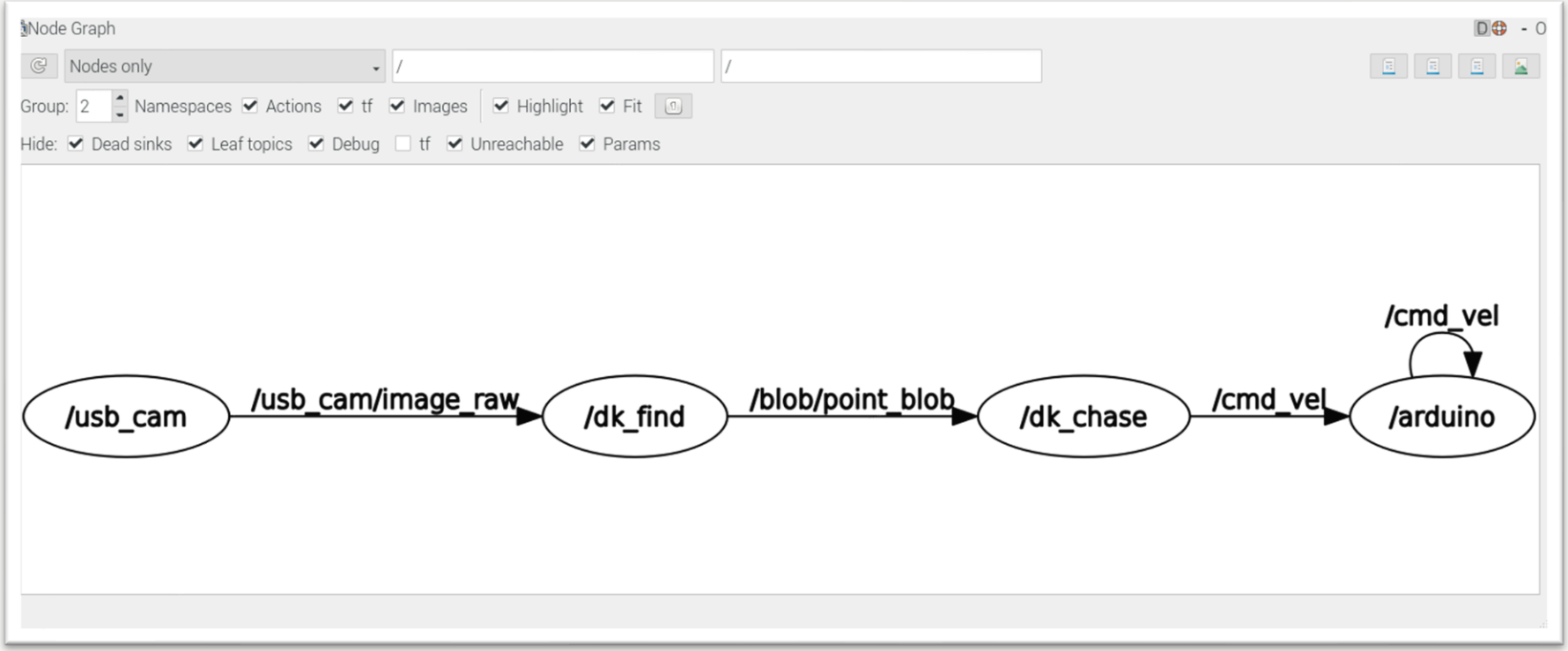


Object detection and recognition

- Detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos



The general overview of notes of this tutorial

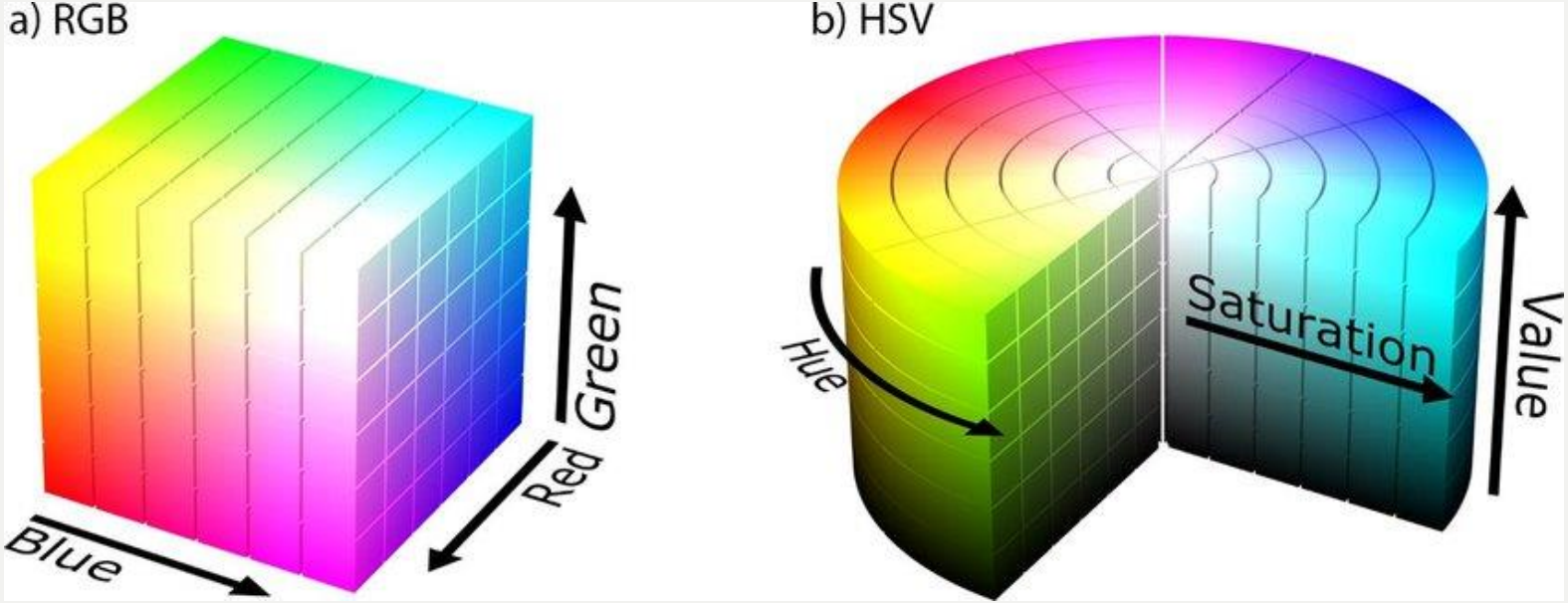


Install the source code for this tutorial

- Go to terminal, type ;

```
:~$ cp ~/catkin_ws/src/
:~$ git clone https://github.com/syahmisanab/blob-chase-bveeta-mini.git
:~$ catkin_make
```

COLOR SPACE



Open cv – prepare sample image

On the terminal, type ;
:~$ sudo apt install cheese

- Take picture of the provided ball save the .jpg
- Move the .jpg to the opencv/include directory

Open cv – blob detection

- Task for participant

- On the terminal
- Go to opencv/include directory
- Run code , python range_detector.py --image <saved image file>.jpg --filter HSV --preview

- Tune the image such as only the ball was shown

Open cv – blob detection

- Task for participant

- On the terminal
- Go to opencv/include directory
- Run code , python range_detector.py --image <saved image file>.jpg --filter HSV --preview

- Tune the image such as only the ball was shown

Open cv – blob detection code explaination

- /home/pi/catkin_ws/src/blob_chase_bveeta_mini/opencv/include/blob_detector.py

blob_detector.py

```
19  import numpy as np;
20
21  #---------- Blob detecting function: returns keypoints and r
22  #-- return keypoints, reversemask
23  def blob_detect(image,                  #-- The frame (cv s
24                  hsv_min,                #-- minimum thresho
25                  hsv_max,                #-- maximum thresho
26                  blur=0,                 #-- blur value (defa
27                  blob_params=None,       #-- blob parameters
28                  search_window=None,     #-- window where to
29                  imshow=False
30                 ):
31
```

Line 23-30

Main function Blob_detect call with ;

Open cv – blob detection code explaination

blob_detector.py

```
30                ):
31
32      #- Blur image to remove noise
33      if blur > 0:
34          image    = cv2.blur(image, (blur, blur))
35          #- Show result
36          if imshow:
37              cv2.imshow("Blur", image)
38              cv2.waitKey(0)
39
```

Line 33-39

We applied blur for denoising the image

```
43      #- Convert image from BGR to HSV
44      hsv     = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
45
46      #- Apply HSV threshold
47      mask    = cv2.inRange(hsv,hsv_min, hsv_max)
48
```

Line 44

Convert the image from BRG to HSV

Line 47

Applied ranging filter that return black and white mask

Open cv – blob detection code explanation

blob_detector.py

```
51          cv2.imshow("HSV Mask", mask)
52
53      #- dilate makes the in range areas larger
54      mask = cv2.dilate(mask, None, iterations=2)
55      #- Show HSV Mask
56      if imshow:
57          cv2.imshow("Dilate Mask", mask)
58          cv2.waitKey(0)
59
60      mask = cv2.erode(mask, None, iterations=2)
61
62      #- Show dilate/erode mask
63      if imshow:
64          cv2.imshow("Erode Mask", mask)
65          cv2.waitKey(0)
```

Line 54-65

Dilate mask and erode mask help removing old particle left over the thresholding

Learn more about eroding and dilating : https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html

Open cv – blob detection code explanation

blob_detector.py

```
74      #- build default blob detection parameters, if none hav
75      if blob_params is None:
76          # Set up the SimpleBlobdetector with default parame
77          params = cv2.SimpleBlobDetector_Params()
78
79          # Change thresholds
80          params.minThreshold = 0;
81          params.maxThreshold = 100;
82
83          # Filter by Area.
84          params.filterByArea = True
85          params.minArea = 30
86          params.maxArea = 20000
87
88          # Filter by Circularity
89          params.filterByCircularity = True
90          params.minCircularity = 0.1
91
92          # Filter by Convexity
93          params.filterByConvexity = True
94          params.minConvexity = 0.5
95
96          # Filter by Inertia
97          params.filterByInertia =True
98          params.minInertiaRatio = 0.5
99
```

Line 75-182

Additional parameter to fine tune the blob detection

Blob_param = none

Learn more about parameter: https://docs.opencv.org/4.x/d0/d7a/classcv_1_1SimpleBlobDetector.html

Open cv – blob detection code explanation

blob_detector.py

```
100     else:
101         params = blob_params
102
103     #- Apply blob detection
104     detector = cv2.SimpleBlobDetector_create(params)
105
106     # Reverse the mask: blobs are black on white
107     reversemask = 255-mask
108
109     if imshow:
110         cv2.imshow("Reverse Mask", reversemask)
111         cv2.waitKey(0)
112
113     keypoints = detector.detect(reversemask)
114
115     return keypoints, reversemask
116
117 #---------- Draw detected blobs: returns the image
118 #-- return(im with keypoints)
```

Line 104
Create blob detector

Line 107
Reverse the mask: black blob on white

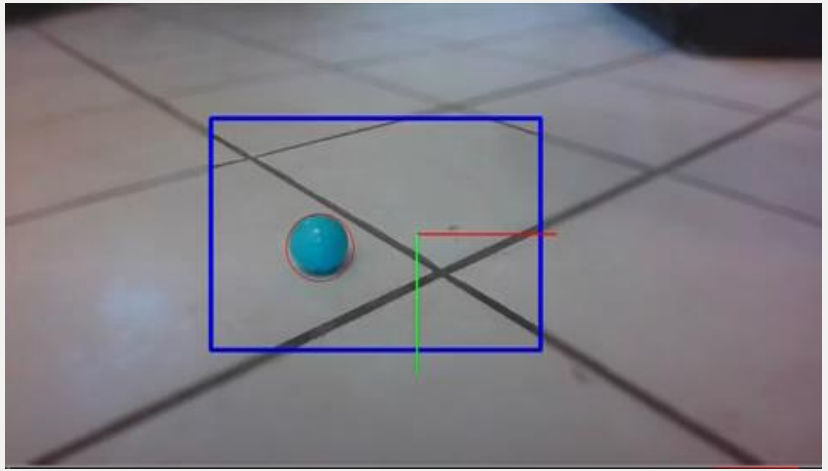
Line 113
pass the reverse mask to the detector to find the position and data of all the detected blobs

Open cv – blob detection code explaination

```
119 def draw_keypoints(image,
120                      keypoints,
121                      line_color=(0,0,255),
122                      imshow=False
123                     ):
```

```
137 def draw_window(image,
138                  window_adim,
139                  color=(255,0,0),
140                  line=5,
141                  imshow=False
142                 ):
```

```
163 def draw_frame(image,
164                  dimension=0.3,
165                  line=2
166     ):
```

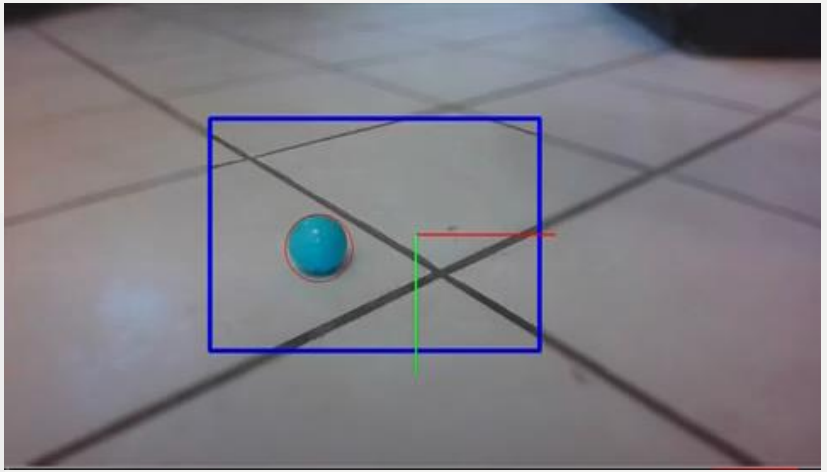


```
185 def apply_search_window(image, window_adim=[0.0, 0.0, 1.0,
186     rows = image.shape[0]
187     cols = image.shape[1]
188     x_min_px    = int(cols*window_adim[0])
189     y_min_px    = int(rows*window_adim[1])
190     x_max_px    = int(cols*window_adim[2])
191     y_max_px    = int(rows*window_adim[3])
192
```

Open cv – blob detection code explaination

```
204 def blur_outside(image, blur=5, window_adim=[0.0, 0.0, 1.0,
205     rows = image.shape[0]
206     cols = image.shape[1]
207     x_min_px    = int(cols*window_adim[0])
208     y_min_px    = int(rows*window_adim[1])
209     x_max_px    = int(cols*window_adim[2])
210     y_max_px    = int(rows*window_adim[3])
211
```

```
225 def get_blob_relative_position(image, keyPoint):
226     rows = float(image.shape[0])
227     cols = float(image.shape[1])
228     # print(rows, cols)
229     center_x    = 0.5*cols
230     center_y    = 0.5*rows
231     # print(center_x)
232     x = (keyPoint.pt[0] - center_x)/(center_x)
233     y = (keyPoint.pt[1] - center_y)/(center_y)
234     return(x,y)
```



Ros node - /dk-find

- /home/pi/catkin_ws/src/blob_chase_bveeta_mini/opencv/src/find_ball.py

blob_detector.py | chase_the_ball.py | find_ball.py

```
129
130             fps = 1.0/(time.time()-self._t0)
131             self._t0 = time.time()
132
133 def main(args):
134     blue_min = (0,45,180)
135     blue_max = (20, 255, 255)
136     # blue_min = (82,31,62)
137     # blue_max = (106, 116, 193)
138     #blue_min = (0,40,183)
139     #blue_max = (8, 255, 255)
140
141     blur     = 5
142     min_size = 10
143     max_size = 40
144
145     #--- detection window respect to camera frame in [x_min
146     x_min    = 0.1
147     x_max    = 0.9
148     y_min    = 0.4
```

- Fine tune the image on the code.

Camera setup

- On the terminal, :~$ [roslaunch usb_cam usb_cam-test.launch]
- On the second terminal, :~$ [rqt_image_view]

- Duplicate usb_cam so that the camera stream at 320*240 resolution

- /home/pi/catkin_ws/src/usb_cam/launch/

Dk chase node – code explaination

- /home/pi/catkin_ws/src/ blob_chase_bveeta_mini/donkey_car/src/chase_the_ball

- chase DK Chase is a node in charge of getting the ball location from the topic blob point blob and publish the throttle and steering angle commands as twist messages to the topic command vel
- the property is detected defines a time out of the which the blob is considered lost
- if the blob is detected the steering angle is calculated as proportional to the blobs horizontal position and the throttle is set to 1
- otherwise, both steering angle and throttle are set to 0

Launch file

- On first terminal, :~$ roscore
- On second terminal, :~$ roslaunch donkey_car blob_chase_demo.launch
- On third terminal, :~$ rqt_image_view